REMARKS

Claims 1-5, 7-13, 15-21 and 23-24 are pending in the present application. By this Response, claims 1-5, 7-9, 11-12, 15, 17-19 and 23 are amended. Claims 1-5, 7-9, 11-12, 15, 17-19 and 23 are amended for clarification purposes. In addition, claims 1, 9 and 17 are amended to recite that the predicate register pair includes two single bit predicate registers. Support for this feature may be found at least at page 18, lines 11-27 of the present specification. Reconsideration of the claims is respectfully requested.

I. Telephone Interview

Applicant thanks Examiner Tsai for the courtesies extended to Applicant's representative during the October 28, 2004 telephone interview. During the interview, the above amendments to the claims were discussed as well as the distinctions of the claims over the cited art. Examiner Tsai indicated that he understood Applicant's position with regard to the distinctions of the claims over the cited art but required additional time to review the cited art before making a final determination as to whether the claims are distinguished over the cited art. The substance of the interview is summarized in the following remarks.

II. Claim Objections

The Office Action objects to claims 3, 4, 7, 8, 11, 12, 17-21, 23 and 24 for a variety of informalities. Each of the phrases in question have been amended for clarity by the present Response. Accordingly, Applicants respectfully request withdrawal of the objection to claims 3, 4, 7, 8, 11, 12, 17-21, 23 and 24.

III. 35 U.S.C. § 112, Second Paragraph

The Office Action rejects claims 1-5, 7-13, 15-21, 23, and 24 under 35 U.S.C. § 112, second paragraph, as being allegedly indefinite for failing to particularly point out and distinctly claim the subject matter, which applicants regard as the invention. This rejection is respectfully traversed.

With regard to claims 1, 7, 9, 15, 17 and 23 have been amended by this Response for clarification to address each of the rejections of these claims. Specifically, claims 1-5, 7-8 have been amended to recite "A process" or "The process" in order to avoid any confusion between the uses of the term "method" in the body of the claims. With regard to claims 7, 15 and 23, these claims have been amended to recite "the try block" and the feature "wherein the snippet invokes a lookup handler for determining if the exception is within the try block of the method" is recited in these claims to recite the feature of the function of determining if the exception is within the try block of the method being performed by a lookup handler invoked by a snippet. In addition, the phrases in claims 3, 11, and 19 that appear to be repetitious have been removed from the claims.

With regard to the phrase "try block" in claims 1, 9 and 17, Applicant respectfully submits that one of ordinary skill in the art is well aware of the definition of the term "try block" as is evident from the description of the try block in the present specification, the Examiner's own citation of the Microsoft Computer Dictionary, and the description of a try block from Sun Microsystems own web site (see attached page from java.sun.com/docs/books/tutorial/essential/exceptions/try.html). A try block is a segment of code that defines the statements or instructions that might throw an exception. Since the term "try block" is known to those of ordinary skill in the art, the claims as they stand are definite. It is not necessary to include a more specific description for the try block in the claims in order for one of ordinary skill in the art to understand the scope of the claims. Thus, claims 1, 9 and 17 are definite.

In view of the above, Applicant respectfully submits that claims 1-5, 7-13, 15-21, 23 and 24 are definite. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 1-5, 7-13, 15-21, 23 and 24 under 35 U.S.C. § 112, second paragraph.

IV. 35 U.S.C. § 102, Alleged Anticipation

The Office Action rejects claims 1, 3, 4, 7-9, 11, 12, 15-17, 19, 20, 23, and 24 under 35 U.S.C. § 102(a) as being allegedly anticipated by Bak et al. (U.S. Patent No. 6,009,517). This rejection is respectfully traversed.

As to independent claim 1, the Office Action states:

> Referring to claim 1, Bak et al.'517 discloses as claimed, a method of handling exceptions in a device (see Fig. 2) having predication, comprising: determining if an exception is pending (see block 955, Fig. 14B, and see also Col. 12, lines 60-62, by checking the local storage) based on values of a predicate register pair (the register files such as 1007, and 1009 in thread local storage, see Fig. 15, are best reasonably and broadly interpreted as a predicate register pair); and handling the exception (by exception handler, see block 905 and block 911 in Fig. 14A) when it is determined that an exception is pending (see block 955, Fig. 14B, and see also Col. 12, lines 60-62, by checking the local storage), wherein handling the exception includes determining if an instruction in a method that threw the exception (see 901, Fig. 14A) is in a try block (the frame, such as JAVA FRAME or C++ FRAME, see Fig. 12, containing the instruction in a method that threw the exception) and invoking a snippet (the frame invoking the exception handler, see also Col. 11, lines 18-25, regarding a JAVA run time system searches an exception handler starting within the function in which the exception was thrown and then propagates through the functions on the execution stack) associated with the method.

Office Action dated July 29, 2004, pages 5-6.

Claim 1, which is representative of the other rejected independent claims 9 and 17 with regard to similarly recited subject matter, reads as follows:

> 1. A process of handling exceptions in a device having predication, comprising:
>
> determining if an exception is pending based on values of a predicate register pair, wherein the predicate register pair includes two single bit predicate registers; and
>
> handling the exception when it is determined that an exception is pending, wherein handling the exception includes determining if an instruction in a method that threw the exception is in a try block and invoking a snippet associated with the method.
>
> (emphasis added)

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicant respectfully submits that Bak does not identically show every element of the claimed invention arranged as they are in the claims. Specifically, Bak does not teach a predicate register pair that includes two single bit predicate registers, as recited in independent claims 1, 9 and 17.

Bak is directed to a mechanism for permitting mixed computing language execution stacks that permits exceptions to be passed down the stack. The Bak mechanism uses a thread local storage to store information about the exception and the return address so that the exception may be passed down the stack which may include stack frames for functions written in multiple programming languages. An example of the thread local storage is shown in Figure 15 of Bak. As shown in Figure 15, the thread local storage includes an exception 1007 and a return address 1009 which allow exceptions to pass from a function written in another programming language, e.g., C++ language, to a Java function, and vice versa (column 13, lines 19-23). Thus, the thread local storage of Bak stores the exception identifier and the return address for the exception.

The Office Action equates the fields 1007 and 1009 in the thread local storage with the predicate register pair recited in independent claims 1, 9 and 17. While there is no teaching or suggestion in Bak that the content of these fields 1007 and 1009 may be stored in predicate registers, let alone a predicate register pair, claims 1, 9 and 17 are amended by this Response to further define the predicate registers recited in these claims. Specifically, claims 1, 9 and 17 are amended to recite that the predicate register pair includes two single bit predicate registers. Bak does no teach or suggest such a feature.

In Bak, an exception identifier and a return address must be stored in order for the exception to be able to be passed down the stack. Neither an exception identifier nor a return address can be stored in a single bit register. A single bit register may have at most two different values – 0 or 1. There is no ability to store an entire exception identifier or entire return address in a single bit register. Thus, the fields 1007 and 1009 of Bak cannot be single bit predicate registers. Therefore, Bak does not teach all of the features of independent claims 1, 9 and 17.

In view of the above, Applicants respectfully submit that Bak does not teach each and every feature of claims 1, 9 and 17 as is required under 35 U.S.C. § 102(a). At least by virtue of their dependency on claims 1, 9 and 17 respectively, Bak does not teach each and every feature of dependent claims 3, 4, 7-8, 11, 12, 15-16, 19, 20, 23, and 24. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 1, 3, 4, 7-9, 11, 12, 15-17, 19, 20, 23, and 24 under 25 U.S.C. § 102(a).

V. <u>35 U.S.C. § 103, Alleged Obviousness of Claims 5, 13 and 21</u>

The Office Action rejects claims 5, 13, and 21 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Bak et al. This rejection is respectfully traversed for at least the same reasons as set forth above with regard to the rejection under 35 U.S.C. § 102(a). Claims 5, 13 and 21 depend from claims 1, 9 and 17, respectively, and thus are allowable over Bak for at least the same reasons as set for the above with regard to claims 1, 9 and 17.

In addition, Bak does not provide any suggestion to utilize a pair of single bit predicate registers or to determine if an exception is pending based on the values of a pair of single bit predicate registers. In fact, Bak teaches away from such a feature in that Bak specifically requires fields in which an exception identifier and a return address must be stored for the system of Bak to function. Such exception identifiers and return addresses cannot be stored in single bit predicate registers as discussed above. Therefore, Bak does not make obvious the features of claims 1, 9 and 17, from which claims 5, 13 and 21 depend, respectively. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 5, 13 and 21 under 35 U.S.C. § 103(a).

VI. 35 U.S.C. § 103, Alleged Obviousness of Claims 2, 10 and 18

The Office Action rejects claims 2, 10, and 18 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Bak et al. in view of Gschwind et al. (U.S. Patent No. 6,513,109). This rejection is respectfully traversed.

Under 35 U.S.C. § 103(c), Gschwind is not prior art that may be properly relied upon in a rejection under 35 U.S.C. § 103(a). Under 35 U.S.C. § 103(c), subject matter developed by another person, which qualifies as prior art only under one or more of subsections (e), (f), and (g) of section 102, shall not preclude patentability where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person. In the present case, the Gschwind reference only qualifies as prior art under 35 U.S.C. § 102(e) and, at the time of the present invention, was owned or subject to an obligation of assignment to the same person, i.e. International Business Machines Corporation. In addition, the present application was filed after November 29, 1999, the effective date of 35 U.S.C. § 103(c).

Therefore, Gschwind is not properly relied upon as prior art and the rejection of claims 2, 10 and 18 under 35 U.S.C. § 103(a) should be withdrawn. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 2, 10 and 18 under 35 U.S.C. § 103(a).

VII. Conclusion

It is respectfully urged that the subject application is patentable over Bak and Gschwind and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE: October 29, 2004

[illegible]

Stephen J. Walder, Jr.
Reg. No. 41,534
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicant

Attachment:

Description of Try Block from Sun Microsystems Web Site

The try Block

The first step in constructing an exception handler is to enclose the statements that might throw an exception within a `try` block. In general, a `try` block looks like this:

```
try {
    Java statements
}
```

The segment of code labelled *Java statements* is composed of one or more legal Java statements that could throw an exception.

To construct an exception handler for the `writeList` method from the `ListOfNumbers` class, you need to enclose the exception-throwing statements of the `writeList` method within a `try` block. There is more than one way to accomplish this task. You could put each statement that might potentially throw an exception within its own `try` statement, and provide separate exception handlers for each `try`. Or you could put all of the `writeList` statements within a single `try` statement and associate multiple handlers with it. The following listing uses one `try` statement for the entire method because the code tends to be easier to read.

```
PrintWriter out = null;

try {
    System.out.println("Entering try statement");
    out = new PrintWriter(
                new FileWriter("OutFile.txt"));

    for (int i = 0; i < size; i++)
        out.println("Value at: " + i + " = " + victor.elementAt(i));
}
```

The `try` statement governs the statements enclosed within it and defines the scope of any exception handlers associated with it. In other words, if an exception occurs within the `try` statement, that exception is handled by the appropriate exception handler associated with this `try` statement.

A `try` statement *must* be accompanied by at least one `catch` block or one `finally` block.